Lista de Exercícios - Structs e Strings

Parte I - Exercícios de Structs

1. Considerando o tipo a seguir (as duas formas de definição de tipo em struct são possíveis - escolha uma delas)

```
typedef struct {
    int horas;
    int minutos;
    int segundos;
}Tempo;
```

ou

```
 struct t{
    int horas;
    int minutos;
    int segundos;
};
typedef struct t Tempo;
```

   Implemente uma **função** que recebe como parâmetro de entrada um valor representando o tempo em segundos e retorna uma estrutura tipo `Tempo`, contendo tais segundos divididos em horas, minutos e segundos.

2. Crie um tipo registro com os campos nome (somente um char indicando a inicial), dia de aniversário e mês de aniversário. Desenvolva um programa que leia do teclado 5 structs e os armazene em um vetor. Depois, mostre para cada um dos meses do ano a inicial do nome dos aniversariantes.

3. Preencha **com dados aleatórios** um conjunto de 15 structs contendo código, telefone da loja e preço de um eletrodoméstico. Desenvolva um programa que permita exibir qual foi a média dos preços cadastrados e uma relação contendo o telefone das lojas cujo preço estava abaixo da média. Obs: os códigos e o telefone devem ser numéricos e devem ser armazenados como int (defina para eles valores que caibam num inteiro).

4. Considere o seguinte esqueleto de um código fonte C:

```
typedef struct {
        float p[3];
        float M;
}Provas;

typedef struct {
        int ra;
        int frequencia;
        Provas Ps;
}Aluno;

int main(void) {
        int n,i,j;
        float desvios_Ps[3]={0.0, 0.0, 0.0}, medias_Ps[3]={0.0, 0.0, 0.0};
        Aluno vetAlunos[MAX_ALUNOS];
     scanf("%d", &n);
    /* COMPLETE AQUI */
}
```

   Complete o esqueleto acima. O programa deve ler um número inteiro n que indicará a quantidade de alunos na turma. Após isso, deve ler as notas das provas de cada aluno, calcular e mostrar na tela as notas e a média de cada aluno, a média geral de cada prova de todos os alunos e o desvio padrão de cada prova de todos os alunos. Note que não é necessaria a declaração de mais nenhuma variável. Exemplo:

```
ENTRADA:
3
030034
4 5 6

123456
0 5 10

987654
6 9 3
```

```
SAIDA:
RA: 030034
Prova 0: 4.000000
Prova 1: 5.000000
Prova 2: 6.000000
Média: 5.000000
----------------
RA: 123456
Prova 0: 0.000000
Prova 1: 5.000000
Prova 2: 10.000000
Média: 5.000000
----------------
RA: 987654
Prova 0: 6.000000
Prova 1: 9.000000
Prova 2: 3.000000
Média: 6.000000
----------------
Média geral P0: 3.333333
Desvio padrão P0: 2.494438
Média geral P1: 6.333333
Desvio padrão P1: 1.885618
Média geral P2: 6.333333
Desvio padrão P2: 2.867442
```

5. Um sistema de administração de pessoal mantém as informações referentes aos funcionários de uma empresa de duas formas: em um vetor de ponteiros para dados estruturados do tipo `Funcionario` ou em um vetor contendo dados estruturados do tipo `Funcionario`. Este tipo é descrito a seguir:

```
struct funcionario {
    int mat; /* matricula do funcionario */
    char nome; /* inicial do nome do funcionario */
    int sup; /* matricula do superior imediato */
};
typedef struct funcionario Funcionario;
```

   (a) Escreva uma função em C que recebe como parâmetros um vetor **contendo estruturas do tipo** `Funcionario`, através do ponteiro `vet`, para seu primeiro elemento, e do inteiro `n`, indicando seu número de elementos, e um inteiro `mat`, representando a matrícula de um funcionário. A função retorna o índice no vetor do funcionário correspondente, ou -1, se este não for encontrado. Protótipo:

   `int Indice_funcionario(Funcionario* vet, int n, int mat);`

   (b) **DESAFIO**: Escreva uma função em C que recebe como parâmetros a variável `vetEnd` que é um **um vetor de endereços** para Funcionario (ou seja um ponteiro de ponteiro de `struct` que equivale a um vetor de ponteiros de struct). Além disso a função também recebe um inteiro `n`, indicando o número de elementos desse vetor de endereços, um inteiro `mat`, representando a matrícula de um funcionário e um ponteiro para inteiro `pTam`. A função deve retornar um novo vetor de estruturas `Funcionario` alocado dinamicamente contendo os funcionários que são subordinados diretos do funcionário cuja matrícula foi recebida por parâmetro. A função também deve retornar por referência (através da variável `pTam`) o tamanho desse vetor de subordinados. Um funcionário no topo da hierarquia, isto é, que não tem um superior imediato, tem o valor -1 no campo `sup`. Por exemplo, suponha que o vetor aponta para as funcionárias Luiza (`mat` = 123 e `sup` = 125), Diana(D) (`mat` = 124 e `sup` = 129), Gina(G) (`mat` = 125 e `sup` = 126), Celia(C) (`mat` = 126 e `sup` = `-1`), Beatriz(B) (`mat` = 128 e `sup` = 126) e Ana(A) (`mat` = 129 e `sup` = 128). Neste cenário, se a matrícula `126` for recebida por parâmetro, a função deve retornar um vetor de estruturas contendo os dados das funcionárias Gina(G) e Beatriz(B) e o tamanho do vetor de subordinados como sendo 2. Protótipo:

   `Funcionario* determinaSubordinado(Funcionario** vet, int n, int mat, int *pTam);`

Parte II - Exercícios de Strings

1. Faça um programa que receba uma frase e conte as vogais, apresentando uma saída tal como a ilustrada abaixo. Exemplo: para a frase `Na proxima quarta-feira eh feriado.`

```
a : ****** (6)
e : *** (3)
i : *** (3)
o : ** (2)
u : * (1)
```

2. Dadas duas cadeias (uma contendo uma frase e outra contendo uma palavra), crie uma função que retorne o número de vezes que a palavra ocorre na frase. Para exemplificar considere a palavra `ANA` e a frase: `ANA e MARIANA GOSTAM DE BANANA`. A palavra `ANA` ocorre 4 vezes. Dica: tente se embasar em funções de busca de sequência em vetores numéricos já implementadas anteriormente.

3. Ao serem fornecidas duas strings, gerar e exibir a intercalação das palavras contidas nas cadeias em uma terceira cadeia. Exemplo:

```
Frase1: Em de espeto de
Frase2: casa ferreiro é pau
Frase3: Em casa de ferreiro espeto é de pau
```

4. Um programa para o registro de usuários de um site da internet gera automaticamente a primeira senha de acesso. Ela é criada a partir de duas cadeias fornecidas pelo próprio usuáio, uma representando o seu nome e outra o bairro onde vive, retirando-se todas as consoantes e espaços dessas cadeias e concatenando-se as duas. Por exemplo, para um usuário que se chama `MARIA LUIZA` e que mora no bairro `REBOUÇAS`, seria criada a senha `AIAUIAEOUA`. Faça um programa que, dadas duas strings representando o nome e o bairro, crie uma senha.

5. Faça um programa que receba um nome completo na forma de uma string e mostre a abreviatura deste nome. Não se devem abreviar as palavras com 2 ou menos letras. A abreviatura deve vir separada por pontos. Ex: Paulo Jose de Almeida Prado. Abreviatura: P. J. de A. P.

6. Refaça a **lista de structs** usando agora **string** no lugar das iniciais das variáveis tipo char. No caso de dados de telefone altere para string de modo a incluir o DDD entre parênteses. Por exemplo: (41)9911223344;